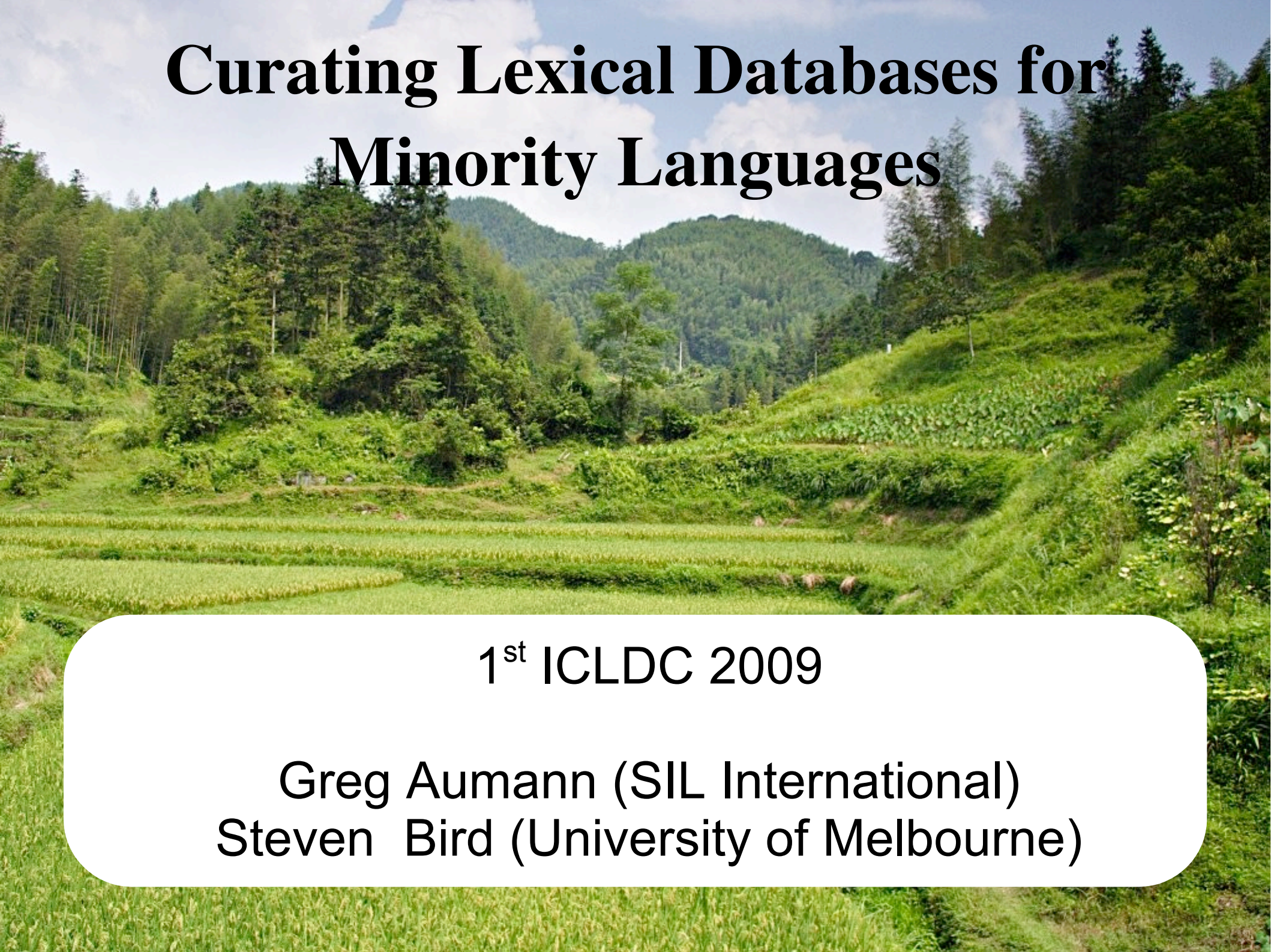


Curating Lexical Databases for Minority Languages

1st ICLDC 2009

Greg Aumann (SIL International)
Steven Bird (University of Melbourne)

Motivations

- Greg
  - proof reading of the Iu Mien—Chinese—English dictionary
  - repurposing Iu Mien—English dictionary
  - hope others benefit from the experience and the code ⇒ this presentation, ⇒ open source code (www.nltk.org)
- Steven
  - Using NLP techniques to support linguistic data management

Curation

- take care of, manage, maintain a resource often for use by others, e.g. curator of a museum
- repurposing data requires structure
- structure requires consistency between elements of the structure
- consistency checking requires structure
- iterative

Scale

- problems
- Our dictionary 10,000
- how long for one proof reading pass?
  - 250 entries per day
  - 40 working days -> 2 months
- after enough proofreading passes cannot see problems easily

Lexical Software

- From less structured to more
  - wordprocessor
  - spreadsheet
  - Toolbox + Multi-Dictionary Formatter
  - Fieldworks Language Explorer
- We are using Toolbox + MDF
  - began in 2001, not many alternatives then

Parsing MDF Entries

- Multi-Dictionary Formatter (MDF)
- complex structure and no validation of structure hence many inconsistencies
- every lexicographer both modifies it and misunderstands it
- using chunking parser from NLTK (Natural Language Toolkit)

Toolbox Record

\lx gomv

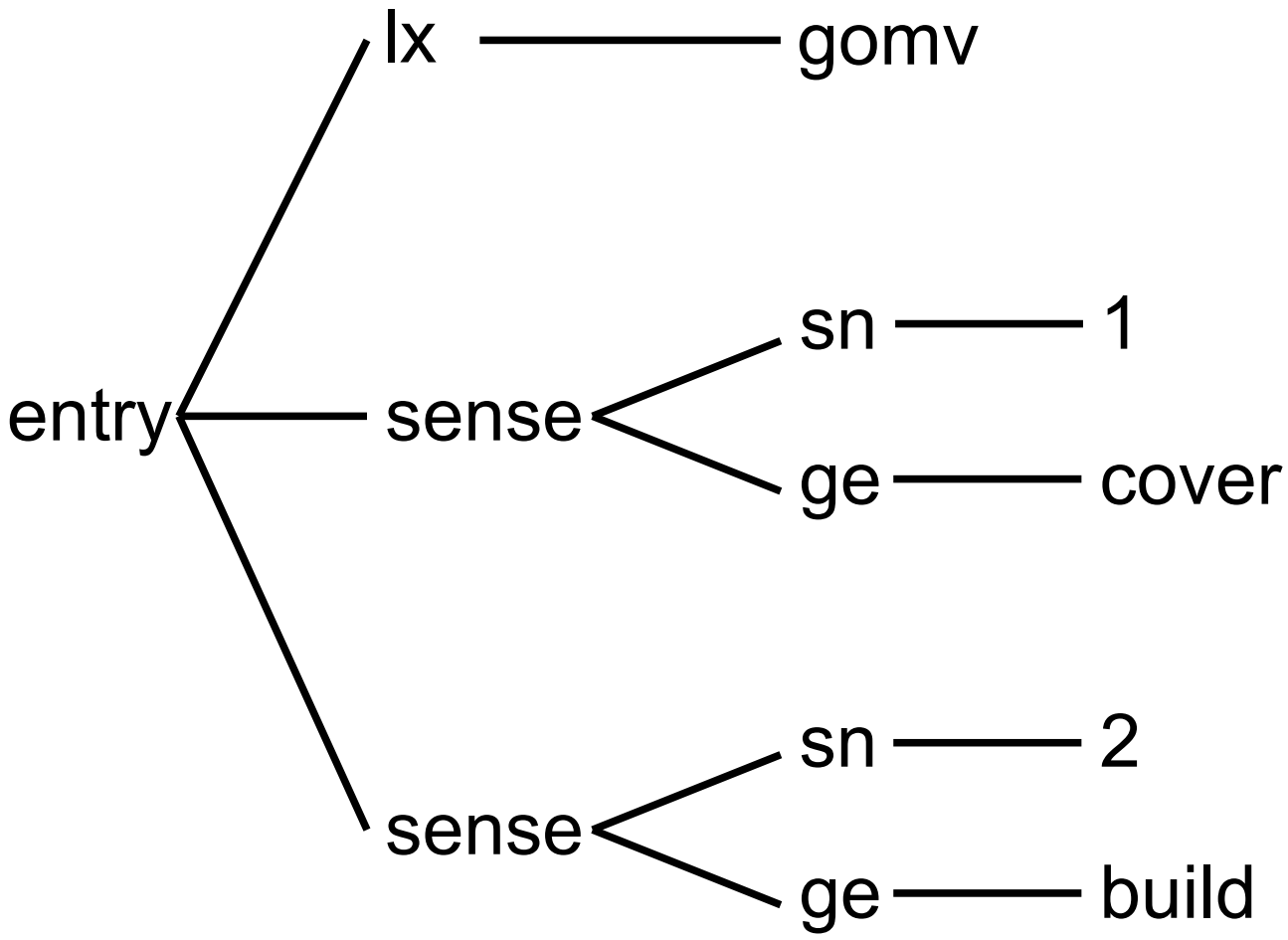
\sn 1

\ge cover

\sn 2

\ge build

Record Structure



Chunk Grammar

sense: {<sn><ge>+}

entry: {<lx><sense>+}

regular expression notes:

+ means 1 or more times

Sense Number Validation

```
lexicon = toolbox.parse_corpus('simple.db', grammar=simple_gram)
for ent in lexicon.findall('record/entry'):
    lexeme = ent.findtext('lx')
    found_sense_nums = []
    for sen in ent.findall('sense'):
        found_sense_nums.append(sen.findtext('sn'))
    num_sens = len(ent.findall('sense'))
    found_error = False
    if num_sens == 1:
        if found_sense_nums != ['']: found_error = True
    elif num_sens == 2:
        if found_sense_nums != ['1', '2']: found_error = True
    elif num_sens == 3:
        if found_sense_nums != ['1', '2', '3']: found_error = True
    if found_error:
        print "lexeme %s: error in sn fields found %s" % (lexeme, found_sense_nums)
```

Validation

- Generate a report of potential problems
- Early attempts likely to turn up parsing problems
- validate frequently
- an iterative process
- Some validation errors solution is obvious
- Other problems will require input from language consultant

Types of Validation

- Overcoming tool shortcomings
  - useful for lexical databases developed using that tool
- Language specific
  - each language in the dictionary e.g. Iu Mien, Chinese and English
  - useful for other dictionaries
- Lexical model specific
  - different even for dictionaries of the same language

Tool Shortcomings

- Toolbox
  - check homonym numbers, sense numbers
  - certain fields occur only once in the entry or sense. e.g. only one \hm in entry, only one \de in a sense etc.
  - targets of cross references and lexical functions exist
  - antonyms have the reverse lexical function

Language Specific

- Check for each language in dictionary
  - spell checking, phonotactics, orthography rules
  - checking for punctuation errors
  - only valid characters used in fields of that language
    - modern tools using unicode are too flexible here
  - checking of fields used for sorting
  - classifier lexical functions point to a classifier entry

Lexical Model Specific

- flexible tools can be too flexible for specific dictionaries
  - e.g. our model is synonyms come in pairs, i.e. simplistic
  - Purnell's model of subentries
  - example sentences contain a '~'
  - entry contains between 1 and 3 semantic domain indexes

Semantic Domains

- All senses indexed with semantic domain
- print lexicon in semantic domain order
- look at all senses in a domain together
  - find misspellings
  - encourages better definitions of words

Conclusions

- Automated validation can save a lot of proofreading effort.
- Essential if planning to import data into more structured lexical tool, e.g. Flex
- Validation useful even in more structured tools
- ability to program empowers the linguist. NLTK toolbox module tries to make this easier for the lexicographer.